

A Feedback Mechanism for Mitigating Denial of Service Attacks against Differentiated Services Clients^{*}

Matthew Braun Geoffrey G. Xie

Computer Science Department
Naval Postgraduate School
Monterey, CA 93940
{[mjbraun](#), [xie](#)}@nps.navy.mil

Abstract

Differentiated Service (DiffServ) networks provide Quality of Service (QoS) guarantees by policing traffic into a fixed number of pre-existing classes. DoS¹ attacks against DiffServ clients will be more targeted and require less attack bandwidth than current attacks due to the per-client and per-class bandwidth limitations which must be imposed to ensure QoS guarantees. In this paper, we present a technique for defeating a DoS attack on a DiffServ client through dynamic modification of packet headers. This technique allows the DiffServ network to distinguish valid traffic from malicious traffic, but does not require cryptographic processing on a per-packet basis and does not increase packet size. We also examine the sensitivity of our system to the traffic policer's token bucket size.

1. Introduction

Differentiated Service provides QoS without maintaining per-flow state information in core routers. Traffic classification is distributed to the edges of the network where volume is lighter. Ingress (edge) routers police traffic

entering the network, classifying and conditioning it to conform to a specific behavior aggregate based on the Service Level Agreement (SLA) between the source and the DiffServ provider. Each behavior aggregate is identified by a single DiffServ code-point, which is stored in the ToS field of the IP header. Within the core of the network, packets are forwarded according to the per-hop behavior associated with the DiffServ code-point [1]. Core routers do not track the state of individual flows. They are only responsible for forwarding based on the marking assigned to each packet when it entered the network.

DoS attacks attempt to artificially exhaust a service provider's resources, such as bandwidth, memory, or processor cycles. Legitimate users are prevented from receiving service due to the lack of available resources. Most DoS attacks rely on the same basic strategy. The attacker compromises a group of non-target hosts, and causes them to send a flood of traffic to the target². The small floods from individual hosts eventually merge into a large flood at the target's upstream router. This flood traffic consumes the bandwidth on the link to the target, causing an overflow of the queue on the link. The source addresses in the packets' headers are usually altered to prevent discovery of the compromised hosts.

Various methods of countering DoS attacks have been proposed. These include Ingress Filtering [3], IP Traceback [5,11], Router Throttling [14], and Distributed Filtering [6]. The drawback common to these methods is the requirement for third-party routers or hosts to cooperate in order for the countermeasure to be effective. Cooperation issues aside, one would ideally wish to counter a DoS flooding attack by stopping it at the source or the source's ISP. Without knowing the true source of the attack, all packets must be treated as valid at the network layer. However, IP alone does not provide a reliable way for the receiver to determine the true source of incoming packets, since the source address field can easily be spoofed.

The Authentication Header (AH) extension to the IP protocol [4] is an established means of verifying the source of a traffic flow. It is possible to determine the validity of a packet's source address for 100% of the packets which use the AH. However, the per-packet cryptographic processing required for IP AH does not scale well, and may be too computationally intensive to implement while maintaining QoS guarantees. Additionally, IP AH requires the

^{*} Research supported in part by DARPA under the Next Generation Internet Program (AO#417) and by National Science Foundation under grant no. ANI-0114014.

¹ In this paper the term Denial of Service (DoS) is used to encompass all forms of denial of service attacks, including distributed (DDoS) and reflected (RDDoS) attacks.

² In the case of a reflected DoS attack, the compromised hosts send their flood traffic to a third party, which (unwittingly) sends a reply to the forged source/target of the flood. This added step is used to further obfuscate the true location of the compromised hosts, and in some cases, to multiply the effective attack bandwidth.

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE OCT 2002		2. REPORT TYPE N/A		3. DATES COVERED -	
4. TITLE AND SUBTITLE A Feedback Mechanism for Mitigating Denial of Service Attacks against Differentiated Services Clients				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Computer Science Department Monterey, CA 93943				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release, distribution unlimited					
13. SUPPLEMENTARY NOTES in Proc. Tenth Int. Conf. on Telecommunication Systems: Modeling and Analysis, pp. 204-213, Monterey, CA, October 2002, The original document contains color images.					
14. ABSTRACT Differentiated Service (DiffServ) networks provide Quality of Service (QoS) guarantees by policing traffic into a fixed number of pre-existing classes. DoS1 attacks against DiffServ clients will be more targeted and require less attack bandwidth than current attacks due to the per-client and per-class bandwidth limitations which must be imposed to ensure QoS guarantees. In this paper, we present a technique for defeating a DoS attack on a DiffServ client through dynamic modification of packet headers. This technique allows the DiffServ network to distinguish valid traffic from malicious traffic, but does not require cryptographic processing on a per-packet basis and does not increase packet size. We also examine the sensitivity of our system to the traffic policers token bucket size.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT SAR	18. NUMBER OF PAGES 10	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

insertion of an additional header field³ into each packet. This can lead to packet fragmentation, which negatively affects QoS.

New types of DoS attacks will accompany implementation of the Differentiated Services model. The separation of traffic into distinct classes and policing traffic on a per client basis will make it easier for an attacker to target a specific subset of the traffic flowing between nodes. Since resources for individual traffic classes will be limited, it may be easier to exhaust resources available to those classes. Furthermore, bandwidth limits imposed on sources in order to maintain QoS guarantees will impose an artificial bottleneck that attackers can exploit. If the DiffServ network reduces the bandwidth available to best effort traffic in order to maintain service guarantees to other traffic, it may inadvertently facilitate a DoS attack against best effort traffic. Service theft, the unauthorized use of guaranteed services, may also result in a denial of service to legitimate users of those services.

Differentiated Service offers new possibilities for the prevention of DoS attacks as well. Since QoS guarantees are only provided to paying clients, the DiffServ provider must maintain a database of clients in order to properly meter traffic and provide appropriate QoS. The provider can use this data at ingress routers to quickly downgrade or drop packets marked with non-client source addresses. Of course, an attacker could simply forge the source addresses of actual clients, so the router must have another means of filtering malicious traffic. However, this requires the attacker to use addresses of hosts that the DiffServ provider knows are valid and can contact to verify the authenticity of the traffic being received.

Based on this observation, we have designed a feedback mechanism through which a provider can notify clients when their traffic does not conform to the profile specified in the SLA. We also have developed a marking method that the client can use in certain cases to make its packets readily distinguishable from traffic with forged headers. The resulting DoS countermeasure

- a) does not require per-packet cryptographic processing,
- b) does not rely on cooperation from third-party hosts or routers,
- c) does not increase the likelihood of packet fragmentation, and
- d) has negligible or no effect on the provider's ability to guarantee QoS.

We have not attempted to devise a solution that will guarantee authentication of 100% of incoming packets or a zero loss rate for valid, in-

³ The minimum size of the AH header is 16 bytes; the maximum size depends on the size of the encrypted packet and the AH options selected

profile traffic. We present the conditions under which our marking method will guarantee 100% authentication, and those under which it will not. For the latter cases, we have derived a formula for predicting the loss rate of valid traffic. We have implemented our solutions in the ns2 network simulator to confirm the correctness of our formula.

The remainder of this paper is organized as follows. Section 2 details the DoS attack scenario that has motivated this work. Our countermeasure for this attack is described in Section 3. The performance analysis and simulation results are presented in Section 4. Section 5 describes several refinements to the countermeasure to enhance its robustness and make it suitable for detection of service theft. Finally in Section 6, we offer some concluding remarks.

2. Attack Scenario

If an attacker could compromise hosts or routers within the DiffServ domain, it could create a DoS for traffic flowing in the domain. Similarly, an attacker could deny service to a client by compromising client systems. While these types of DoS are both possible and effective, solutions to them are beyond the scope of this paper. Our research focuses on a method of attack similar to the most common types of attacks employed in the current Internet. Below, we set out the conditions under which this type of attack is possible.

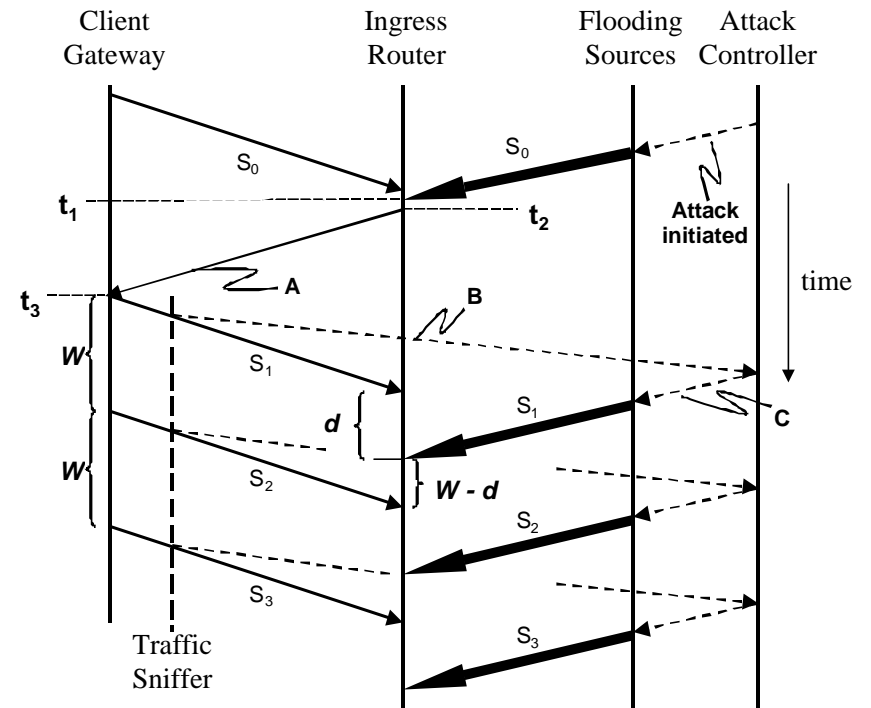
A bandwidth consumption attack is harder to accomplish if the flood traffic must traverse a DiffServ network. The network will give preferential treatment to packets from paying clients. Packets from unrecognized sources will be assigned a default code-point, which equates to best effort service. Valid packets will be more likely to reach their destination because of the priority they receive over best-effort traffic. While this will provide more DoS protection for the endpoints of flow paths, the overall effect will be a shift in the focus of DoS attacks to the point at which the flow enters a DoS domain. This is the logical point to attack the traffic since bandwidth limitations imposed by SLA enforcement will make the DiffServ network the narrowest section of the path.

DiffServ domains only provide preferred service to recognized clients. At the network layer, incoming packets are classified to receive preferential treatment if their source address matches the address associated with an existing SLA. Consequently, an attacker must mark malicious packets with the address of a valid DiffServ client to attack the class of traffic associated with that client or SLA.

The DiffServ domain must be able to meter each client's traffic in order to ensure client adherence to SLAs with regard to usage amounts. It is

In a wired network, if a client is only one hop away from the ingress router, the DiffServ domain will also be able to filter traffic based on the incoming link. In this case, it will be impossible for an attacker to flood spoofed traffic using this client's source address, since we have already stated that the client itself cannot be compromised. It follows that no attack is possible unless the client is more than one hop away from its assigned ingress router. This does not hold true, however, if the client's connection to the ingress router is a wireless link, since the transmission medium itself is not secure.

Figure 1 is a time diagram showing the sequence of actions involved in our proposed countermeasure. When the ingress router marks a packet that appears to originate from a DiffServ client as out-of profile, it will log the source and time (t_1) of the drop. When the rate of out-of-profile marking exceeds a pre-set threshold (t_2), the router will send a feedback message (A) to



The router feedback to the client will consist of a router-generated seed key for an algorithm that generates a sequence of signatures. The client and router will be able to independently calculate what the correct signature should be using this algorithm and the seed-key. The algorithm can be well known as long as the seed key being used remains secret. The seed key will be encrypted using a shared secret key and digitally signed. The seed key is used to generate new signatures instead of the shared secret key to avoid compromising the secret key through overuse. The digital signature provides authentication for the feedback message, so attackers will be unable to create a DoS by forging

these messages. Payload encryption is required since the attacker can monitor traffic flowing between the client and the DiffServ domain. Since the algorithm for generating signature values is not secret, access to unencrypted seed keys, would allow the attacker to change the signature of the attack packets as rapidly as the sender could, thus circumventing the countermeasure.

Upon receipt of a feedback message, the client will authenticate it, decrypt the payload, and use the seed key to calculate the sequence of signature values that it will use. The client will immediately begin using the values in the designated fields of the IP headers of its packets. It will switch to the next signature (S_i) in the sequence at regular intervals denoted by W . The attacker will not know what each new signature is until it receives the information from the monitor installed along the path of the client traffic (B). When it knows the new signature, it can direct the flood sources to change the signatures they are using (C). The time between when the ingress router receives the first valid packet with a new signature and when it receives the first attack packet with the same signature, denoted by d , is the window in which 100% authentication is possible. The importance of the relative values of d and W are discussed in section 4.

The seed key is also used to create the same sequence of signature values at the router. After sending a feedback message, the router will treat all packets as valid until it receives the first packet with the first altered signature. All successive packets with an incorrect signature are dropped, except the first packet received with the second signature. When the first packet with the second signature is received, the router will drop all successive packets that do not match the second signature, including those marked with the first signature. This prevents an attacker from using old signatures to circumvent the DoS countermeasures.

4. Performance Evaluation

In this section, we try to evaluate the effectiveness of the feedback mechanism in mitigating the DoS attack. The main performance metric of interest is the *client's packet out-of-profile rate*, i.e., the percentage of the DiffServ client's packets being marked out of profile at the ingress router. We consider a DoS countermeasure more effective than another if it achieves a smaller rate of out-of-profile packets for the client given the same network setup and attack scenario. As a secondary interest we also examine the fairness aspect of the feedback mechanism. Obviously it is desirable that a DoS countermeasure be able to distribute out-of-profile packets evenly among all current connections of the DiffServ client. Finally, we are also interested in how the feedback

mechanism may aid in the detection of DoS attacks and the more elusive service thefts. We will treat this important topic separately in Section 5.2.

Our performance evaluation consists of two steps. In the first step, we create an analytical model of the feedback mechanism with a set of simplifying assumptions and then derive from this model a closed form solution for the DiffServ client's packet out-of-profile rate. In the second step, we verify the analytical results via simulation experiments.

4.1. Derivation of Client's Packet Out-Of-Profile Rate

Denote the percentage of the client's packets being marked out of profile at the ingress router by p . In order to derive p , we make the following additional assumptions:

1. All packets have the same size.
2. The client's traffic arrives at the ingress router at a constant rate of r packets per second, which is less than or equal to CIR , the client's allocated committed information rate in packets per second.
3. The attack traffic arrives at the ingress router at a constant rate of A packets per second such that

$$r + A \geq CIR \quad (1)$$

This means the percentage of valid traffic received by the ingress router is equal to

$$\frac{r}{r + A} \quad (2)$$

4. The client traffic switches to a new signature every W seconds. The attack traffic tries to make the same signature change, but the change always happens d seconds later from the ingress router's perspective. In other words, there is a fixed lag of d between the arrival time at the ingress of the first valid packet with a new signature and the arrival time of the first attack packet with the same signature.
5. The ingress router's traffic metering process for the client is fair so that if the traffic being metered is made of several flows, each flow will be ensured of a share of in-profile packets that is proportion to the flow's packet arrival rate.

Consider the time window for an arbitrary signature used by the client's traffic. There are two cases:

- Case 1: $W \leq d$. From assumption 4, during the entire time period, every attack packet carries an expired signature when inspected by the ingress router. Such packets will be dropped before being counted against the client's committed rate in the metering process. From assumption 2, the rate of the valid traffic alone does not exceed the committed rate. Thus, we have $p = 0$.
- Case 2: $W > d$. From assumption 4, during an initial time period equal to d , the ingress router will be able to drop all attack packets. However, for the remaining time of $W - d$, the ingress will not be able to distinguish valid traffic from attack traffic because they have the same signature. In that case, some of the client's packets will be marked out-of-profile. From assumption 5 and Equation (2), the percentage of client packets marked as in-profile during this period is:

$$[(W - d) \cdot CIR] \frac{r}{r + A}. \quad (3)$$

So the number of client's packets marked out of profile during this period is:

$$r \cdot (W - d) - [(W - d) \cdot CIR] \frac{r}{r + A}, \quad (4)$$

$$= (W - d) \cdot r \times (1 - \frac{CIR}{r + A}). \quad (5)$$

Dividing (5) by the total number of packets sent by the client during the entire time window, $r \cdot W$, we obtain:

$$p = (\frac{W - d}{W}) \times (1 - \frac{CIR}{r + A}). \quad (6)$$

It can be shown via a similar derivation that p_0 , the packet out-of-profile rate for a client that does not use any DoS countermeasure, is equal to

$$p_0 = (1 - \frac{CIR}{r + A}). \quad (7)$$

Using equation (7), we rewrite equation (5) as:

$$p = (1 - \frac{d}{W}) \times p_0. \quad (8)$$

Equation (8) clearly indicates that the reduction in the packet out-of-profile rate due to the feedback mechanism is inversely proportional to W , the period between signature changes by the client.

Combining both cases, we have proved the following theorem.

Theorem 1. If assumptions 1-5 hold, then after the client initiates the DoS countermeasure as a result of the feedback mechanism, the client's packet out-of-profile rate becomes

$$p = \max\{0, (1 - \frac{d}{W}) \times (1 - \frac{CIR}{r + A})\}. \quad (9)$$

4.2. Fairness

The DiffServ client's traffic is typically an aggregation of packets from a number of application flows. Theorem 1 gives the packet out-of-profile rate for the aggregate. One can use it to predict the *total* number of client packets marked out of profile in any given time period. However, Theorem 1 does not tell how these out-of-profile packets are distributed among the flows. In other words, we need to examine further how fairly each flow is treated by the DoS countermeasure. It is important that the out-of-profile packets be spread out evenly. The application associated with an individual flow, such as a TCP connection or a Voice-over-IP (VoIP) receiver, is usually designed to resist a limited amount of QoS degradation to the flow's packets. If the QoS degradation exceeds that limit, as in the case where the flow has a disproportionately large number of packets marked out-of-profile, the application may not be able to function properly.

Denote the set of flows carried by the client traffic by S . Let $p(f)$ be the packet out-of-profile rate of flow f . We measure the fairness of a DoS countermeasure by the following metric, called *fairness index*:

$$F = \max_{f \in S} \{p(f)\} - \min_{g \in S} \{p(g)\}. \quad (10)$$

Clearly the value of F has a range between 0 and 1. The closer F is to 0, the smaller the maximum difference between any two flows' packet out-of-profile rates is. Therefore, it is important for a DoS countermeasure to minimize F .

Next we evaluate the fairness property of the DoS countermeasure based on our feedback mechanism. It is possible to develop an analytical solution for F by making additional assumptions on each flow's packet arrival process. However for brevity, we will skip such a formal treatment. Instead, we try to qualify the fairness performance of the DoS countermeasure with the following observation.

Recall the steps for deriving the client's out-of-profile rate. If W is a constant and if W is larger than d , the DoS countermeasure creates periodic DoS effective time intervals as illustrated in Figure 2 below.

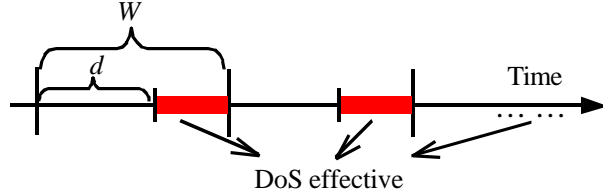


Figure 2. Periodic DoS Effective Time

During these time intervals, the countermeasure is ineffective and the DoS attack causes high packet out-of-profile rates for the client. Now suppose that one of the client's flows, e.g., created by a NetMeeting application, generates packets periodically. It is possible that the flow started during a DoS effective interval and its packet generation period is similar to W . In such a case, the flow's packets always arrive at the ingress during DoS effective periods, resulting in a disproportionately high out-of-profile rate for the flow.

Therefore, the DoS countermeasure may not treat the flows fairly. One fix is to have the client randomize the value of W . We intend to evaluate the performance of this fix and other solutions to enhance the fairness of the countermeasure in our follow-on work

4.3. Experimental Results

In this subsection we describe the simulator, the extensions that were made to the simulator to implement our countermeasure, the experimental network topology used during simulation, and the simulation results.

4.3.1. Simulator and Extensions

The experiments were conducted using version 2.1b8a of ns2, a discrete event simulator targeted at networking research [6]. The simulator is object-oriented, written in C++, and uses Otcl as a command and configuration interface [2]. The simulator includes a DiffServ module with implementations of distinct core and edge routers, several marking policies, and built-in tracing for DiffServ queues.

The DoS countermeasure was implemented through creation of new objects inherited from existing ns2 objects. Significant changes include modification of the existing policing algorithm employed by DiffServ edge routers, implementation of a means of periodically altering the IP ToS field at run-time, and creation of a new agent to carry the feedback messages from the DiffServ ingress router to the client. Several Otcl functions were also added to provide necessary links between the command interface and the C++ classes.

After examining the IP header format, we have chosen to use the Type of Service (ToS) field in the IP header as the mutable portion of the packet signature in our simulation. The ToS field is unused in the non-DiffServ routers between the client and the DiffServ provider, so modifying it at the source will not affect packet routing outside of the DiffServ domain.⁴ DiffServ ingress routers change this field after receipt based on the client's SLA, so modifying it will not affect routing within the DS domain. In the remainder of this section, signature refers specifically to the combination of IP source address and IP ToS field. However, other fields such as ID or Options could be used in place of or in combination with the ToS field to determine packet signature.

4.3.2. Topology

Figure 3 shows the ns2 topology that was used to conduct experiments. The times t , t_A , t_C , and t_F represent the sum of all delays incurred by a packet transiting the respective link, including processing, transmission, and propagation delays. We observe that the difference in arrival times at the ingress router of the first valid and invalid packets with the same signature, which we have previously named d , can be written

$$d = (t_A + t_C + t_F) - t. \quad (11)$$

For all simulations the client and flood sources were set to transmit fixed sized packets at a constant bit rate. A small degree of random variation in packet

⁴ In practice we would not choose to use this field exclusively, since it may be used in transit by networks that implement IP Precedence [10].

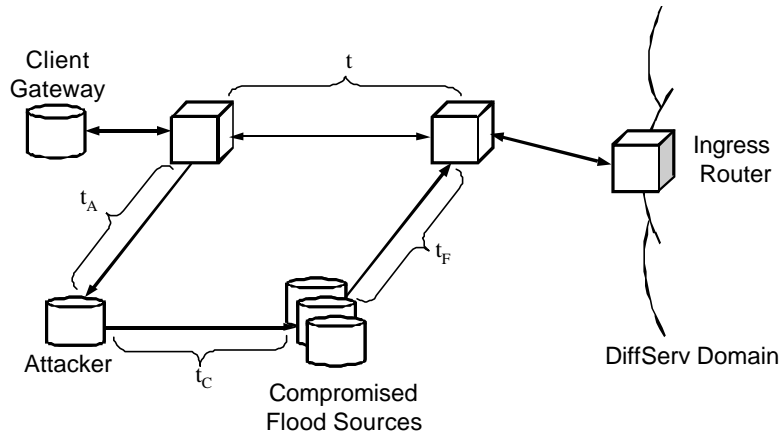


Figure 3. Experimental Network Topology

inter-departure times was introduced to eliminate synchronization of packet arrival at the ingress router. The ingress router used a Token Bucket policing method to assign code points to incoming packets.

4.3.2. Experiments

In our first set of experiments, we compared the out-of-profile rate produced in the simulation to that calculated using Equation (9). Runs were conducted for several different values of p_0 . The values of W , and p_0 were held constant during each run, and the value of d was manipulated by varying t_A while holding t_C , t_F , and t constant. The results of these trials are shown in Figure 4.

The simulated results correlated well with our predicted results. For cases in which $W > d$, but the difference was small, the countermeasure was effective in limiting the out-of-profile rate for valid packets. When $W \gg d$, the out-of-profile rate for valid packets approached p_0 .

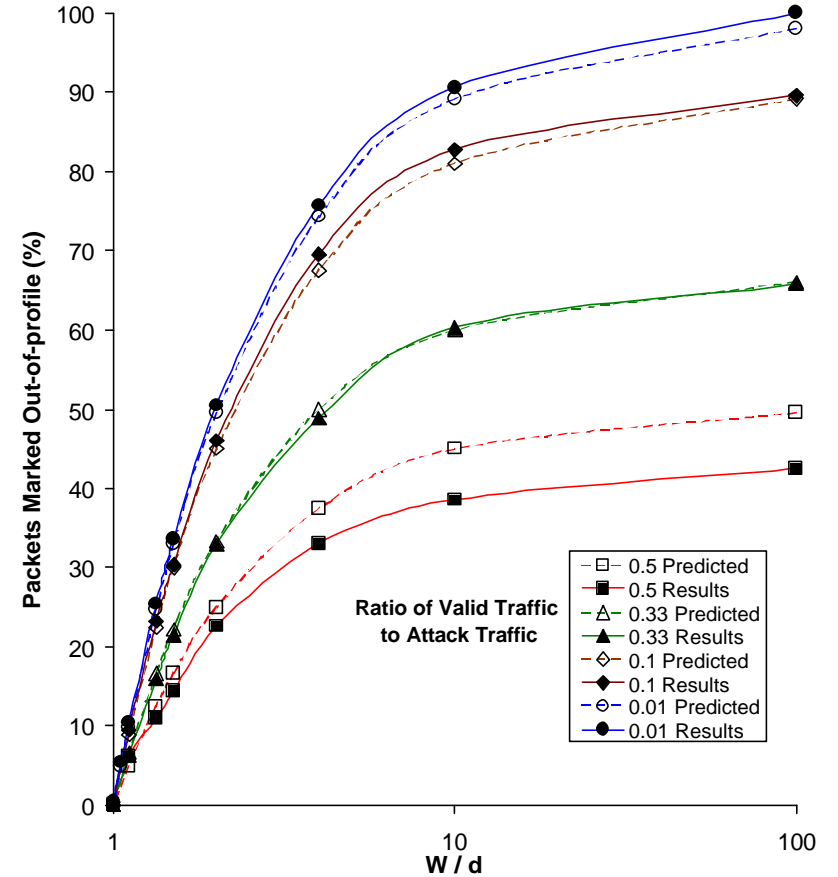


Figure 4. Predicted Out-of-Profile Rates vs. Measured Results

Our secondary interest was to observe the effect changing the bucket size for the Token Bucket policer would have on the DoS countermeasure. In each run, the values of W , d , and p_0 were held constant. The size of the token bucket was increased exponentially until it was large enough to prevent any packets from being dropped regardless of their true source. Runs were conducted for cases in which $W > d$ and $W < d$. The results are plotted in Figure 5. They indicate that for our countermeasure, a small token bucket size is required

to minimize out-of-profile marking for valid traffic while maximizing it for invalid traffic. We also note that the marking rate for valid traffic is higher than predicted if the token bucket size is not optimized. We attribute this to the longer delay in starting the countermeasure that logically accompanies a larger token bucket. If the ingress router allows larger bursts of traffic, then it will take longer for packets to be marked out-of-profile once an attack commences.

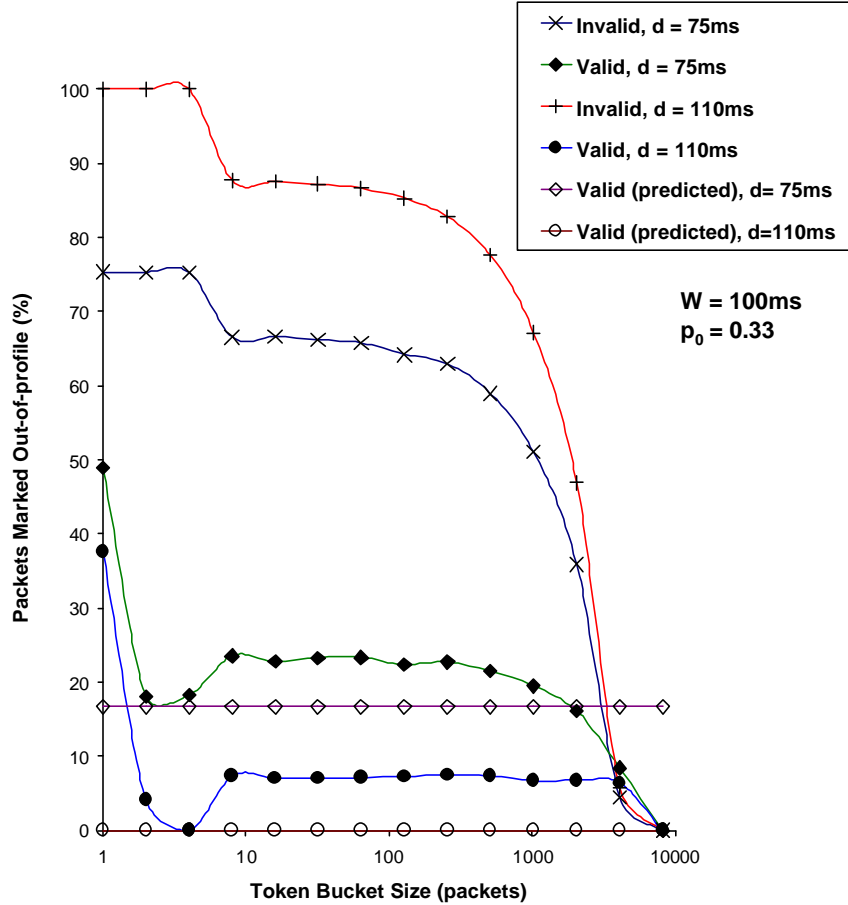


Figure 5. Effect of Token Bucket Size on Out-of Profile Rate

5. Discussion

In this section, we point out some potential weaknesses of a naïve implementation of the proposed DoS countermeasure algorithm and describe how to refine the algorithm to eliminate them. We also discuss ways of extending the algorithm to aid in detection of service theft.

5.1. Algorithm Refinements

So far in the discussions, we have assumed that the attacker behaves in a predictable way. In reality, however, the attacker will be much less predictable. One sure thing is that the attacker will try to find holes in the DoS countermeasure algorithm itself and attack them directly. It is important to anticipate such attacks and refine the algorithm to fix these holes.

One direct attack on the DoS countermeasure as we have implemented it can be launched as follows. The attacker contrives its attack traffic so that every possible ToS value is used at a high frequency. Since there are a total of only 256 different ToS values, it is not difficult for the attacker to do so. As a result, it occurs frequently that an attack packet carries a ToS matching the next ToS expected by the ingress router. Processing that packet will trigger a premature signature change at the ingress. The resulting signature asynchrony between the client traffic and the ingress router will cause all future client packets to be dropped by the DoS countermeasure.

The above scenario actually does more damage to the client than the target scenario described in Section 2. To deal with this problem, the DoS countermeasure algorithm must be refined to include a reliable method for the client traffic and the ingress router to synchronize their signature updates. For this paper, we first examined a solution based on the concept of time-driven key sequencing [13]. The client and the ingress router both update their signatures periodically and each uses its own local clock to determine the update instances. This method has the advantage of low communication overhead because it requires no additional message exchange between the client and the ingress router. However, it requires tight synchronization between the clocks of the client and the ingress router. Unfortunately, achieving tight clock synchronization in a secure fashion without using specialized hardware (e.g., GPS clocks) remains an open problem.

When tight clock synchronization between the client and the ingress router is not possible, we propose to refine the DoS countermeasure algorithm as follows. The client sends a special signaling packet to request the ingress router begin accepting packets marked with the next signature in the sequence.

Each signaling packet is marked with the signature in use prior to the update so that it will not be dropped due to the DoS countermeasure. Like the signatures, the sequence of payloads of these signaling packets, e.g., a sequence of 64-bit values, may be pre-computed⁵ at both the client and the ingress router based on the seed key and the shared secret. Therefore, their creation incurs little extra latency to the data path of normal client traffic. The ingress router checks the validity of a signaling packet based on the packet's payload. A valid signaling packet should carry the next payload expected by the ingress router. The large number of bits in the payload makes it virtually impossible for the attacker to succeed in producing a valid signaling packet with a brute-force approach.

The signaling packets may be identified in several ways. For example, the client may put an all-zero Identification field in their IP header. For reliability, the client may send multiple signaling packets for each signature update. These packets will not be metered against the client's committed rate, and once the change has been made, any subsequent signaling packets for the same update will be automatically discarded, since they are marked with the previous signature. It is not necessary for a signaling packet to deliver the new signature since the ingress router has pre-computed all the signatures. On the other hand, if the new signature is always appended to the signaling packet payload, then there is no need for the ingress router to pre-compute the signatures.

Another problem arises when the client becomes idle for a long time. Without new client traffic to trigger a change, a signature may remain valid at the ingress router for a very long time, opening a door for service theft. A simple solution to this problem is having the ingress router time out a signature if it has not received a new signaling packet for a predetermined amount of time.

It should be noted that these refinements for achieving better signature synchronization between the client and the ingress router have very little impact on the performance analysis presented in Section 4.

5.2. Detection of Service Theft

A service theft can be defined as a course of actions taken by a perpetrator to use a portion of a valid client's allocated bandwidth and obtain a premium service without pay. Unlike a DoS attack, the intruder is typically much more restrained to cover his tracks. Therefore, a service theft usually does not cause as much direct harm to the client as a DoS attack. For example, the intruder

⁵ For example, the new signature and related signaling packet payload may be extracted from one 128-bit keyed-MD5 hash value.

may study the client's traffic pattern and adjust his own traffic volume over time so that the ingress router will never mark an excessive number of packets out of profile for that client. This service theft results in lost revenue and network availability for the DiffServ provider. Therefore, the service provider will treat them as serious offenses that must be dealt with, in the same way cable companies handle illegal cable installations.

The proposed feedback mechanism and the resulting cooperation between the client and the ingress router may help the service provider detect service theft. For example the ingress router may activate the feedback mechanism randomly, regardless of whether or not it has just marked a large number of packets out of profile for the client. It is important for the system to use a random signature time window (W) so that the intruder is unable to predict how much longer a spoofed signature may be valid and adjust its traffic pattern accordingly. If a service theft is under way, the ingress router should notice two or more signatures being frequently used at the same time. When this occurs, the ingress router may log the event as a possible occurrence of service theft or immediately alert the network operator to perform further investigations.

If the intruder can monitor feedback messages from the ingress router, he can defeat the ingress router's service theft detection by suspending his flooding traffic upon seeing a feedback message. In other words, the proposed theft detection mechanism may not catch service theft launched by a more resourceful intruder. However, it will be sufficient to stop the service theft from continuing, and more importantly, it will deter future service theft by forcing the intruder to expend more resources and effort to avoid detection.

A general conclusion can be drawn from the above discussion. That is, the proposed DoS countermeasure may be extended into an auditing function that is orthogonal to access control. For example, it may be used in conjunction with an IP AH based packet authentication scheme to mitigate DoS attacks and detect service theft in cases where the authentication process has been compromised.

6. Concluding Remarks

We have demonstrated that it is possible to mitigate DoS attacks against DiffServ clients and detect service theft without per-packet cryptographic processing. The tradeoff for the efficiency gain is the lack of guarantee on 100% rejection of malicious packets at the ingress. Our view is that DoS attacks and service theft are more of a QoS guarantee problem than one about security assurance. The proposed countermeasure should be combined with other security protocols if both QoS guarantee and security assurance are required.

Its low cost makes it an excellent choice as an independent monitor for possible breaches of the security protocols.

We predict that some form of feedback mechanism is going to be installed for DiffServ clients in the future to take advantage of the explicit bond (SLA) between the client and provider. If that happens, it will take very little effort to deploy the proposed DoS countermeasure.

Finally, we would like to point out that the proposed scheme might also be used to mitigate DoS attacks against a client that are launched inside the DoS domain. In this case, the egress router will have to be the one who sends a feedback message to the client and filters out malicious packets based on dynamic packet signature.

References

- [1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss. An Architecture for Differentiated Services, RFC 2475, IETF, December 1998.
- [2] K. Fall and K. Varadhan, eds. The ns Manual. February 2002. Available at <http://www.isi.edu/nsnam/ns/ns-documentation.html>.
- [3] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing. RFC 2827, May 2000.
- [4] S. Kent and R. Atkinson. IP Authentication Header. RFC 2402, IETF, November 1998.
- [5] H. Lee and K. Park. On The Effectiveness of Probabilistic Packet Marking for IP Traceback Under Denial of Service Attack. *Proceedings of IEEE INFOCOM 2001*, Anchorage, Alaska. April 2001.
- [6] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling High Bandwidth Aggregates in the Network. Technical Report (Extended Version), ACIRI and AT&T Labs Research, July 2001.
- [7] The Network Simulator – ns2. <http://www.isi.edu/nsnam/ns/>
- [8] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, IETF, December 1998.
- [9] K. Park and H. Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. *Proceedings of ACM SIGCOMM 2001 Conference*, pp. 15-26, Zurich, Switzerland, September 2001.
- [10] J. Postel, Editor, Internet Protocol. RFC 791, IETF, September 1981.
- [11] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. *Proceedings of ACM SIGCOMM 2000 Conference*, pp. 295-306, Stockholm, Sweden, August 2000.
- [12] J Stephen. The Changing Face of Distributed Denial of Service Mitigation. White Paper, August 2001. Available at <http://www.sans.org/infosecFAQ/threats/face.htm>.
- [13] G.G. Xie, T. Levin, and C Irvine. Quantify the Effect of Clock Drift and Network Latency on Time-driven Key Sequencing. *Proceedings of 22nd International Conference on Distributed Computing Systems – Workshop on Assurance in Distributed Systems and Networks*, Vienna, Austria, July 2002.
- [14] D. Yau, F. Liang, and J. Lui. On Defending against Distributed Denial-of-service Attacks with Server-centric router throttles. Technical Report, CERIAS and Department of Computer Science, Purdue University. May 2001.
- [15] F. Zhi, S. F. Wu, T.S. Wu, He Huang, F. Gong. Security Issues for Differentiated Service Framework. Internet Draft (expired) draft-fu-diffserv-security-00.txt, IETF, October 1999.

Matthew Braun is an officer in the United States Navy. He holds a BS in Engineering Mechanics from the University of Illinois. At the time of writing, he was completing his MS in Computer Science at the Naval Postgraduate School in Monterey, CA. His research interests include network security and network Quality of Service.

Geoffrey G. Xie received the B.S. degree in Computer Science from Fudan University, Shanghai, China in 1986. He received the M.S. degree in Computer Science and the M.A. degree in Mathematics from Bowling Green State University, Ohio in 1988, and the Ph.D. degree in Computer Science from the University of Texas at Austin, Texas. From 1991 to 1993, He worked full-time as a project engineer in Schlumberger Systems Center in Austin, Texas. He joined the Department of Computer Science of the Naval Postgraduate School in 1996 and is currently an Associate Professor in that department. He serves on the editorial board of the Computer Networks Journal. His research interests are network management, traffic engineering, network security, and mobile ad hoc networks. He is a member of IEEE.